Gedeon_Muhawenayo

June 15, 2020

1 Building an Object Tracker

Tracker detects objects in all frames of a video and link the predictions from one frame to the next.

Imports and installation of necessary packages

```
In [1]: !pip install -U torch==1.5 torchvision==0.6 -f https://download.pytorch.org/whl/cu101/t
        !pip install cython pyyaml==5.1 >/dev/null
        !pip install -U 'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI
        import torch, torchvision
        #print(torch.__version__, torch.cuda.is_available())
        #!gcc --version

        # mask r-cnn
        !git clone https://github.com/matterport/Mask_RCNN.git > .null
        !pip install -r Mask_RCNN/requirements.txt > .null
        !pip install ./Mask_RCNN > .null

        # install detectron2:
        !pip install detectron2==0.1.2 -f https://dl.fbaipublicfiles.com/detectron2/wheels/cu10

        # Setup detectron2 logger
        import detectron2
        from detectron2.utils.logger import setup_logger
        setup_logger()

        # import some common libraries
        import os
        import random
        import time
        import matplotlib.pyplot as plt
        import numpy as np
        import cv2
        from google.colab.patches import cv2_imshow
        from cv2 import VideoWriter, VideoWriter_fourcc, imread, resize
        from mrcnn.visualize import display_instances
        from google.colab import files
        # import some common detectron2 utilities
        from detectron2 import model_zoo
        from detectron2.engine import DefaultPredictor
        from detectron2.config import get_cfg
        from detectron2.utils.visualizer import Visualizer
        from detectron2.data import MetadataCatalog
```

```
  Running command git clone -q https://github.com/cocodataset/cocoapi.git /tmp/pip-req-build-ep
Cloning into 'Mask_RCNN'...
remote: Enumerating objects: 956, done.
remote: Total 956 (delta 0), reused 0 (delta 0), pack-reused 956
Receiving objects: 100% (956/956), 116.75 MiB | 13.31 MiB/s, done.
Resolving deltas: 100% (567/567), done.
```

1.1 Part A: Detecting Objects in Frames

1.1.1 Data; Downloading the video clip

```
In [1]: !wget https://github.com/gkioxari/aims2020_visualrecognition/releases/download/v1.0/vid
        !unzip videoclip.zip >/dev/null
```

1.1.2 Pretrained model for semantic segmentation

```
In [3]: cfg = get_cfg()
        cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50
        cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7  # set threshold for this model
        cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R
        seg_predictor = DefaultPredictor(cfg)
```

```
model_final_f10217.pkl: 178MB [00:17, 10.3MB/s]
```

1.1.3 Prapare images for semantic segmentation

```
In [0]: def read_images(folder = 'clip'):
            images_list = []
            for file_name in sorted(os.listdir(folder)):
              image = cv2.imread(folder+'/'+file_name)
              images_list.append(image)
            return images_list

        images_list = read_images(folder = 'clip')
```

1.1.4 Visualizing samples of the loaded images

```
In [5]: for indx in random.sample(range(0, 40), 3):
          cv2_imshow(images_list[indx])
```

Florian
Bad Schwartau
GRS

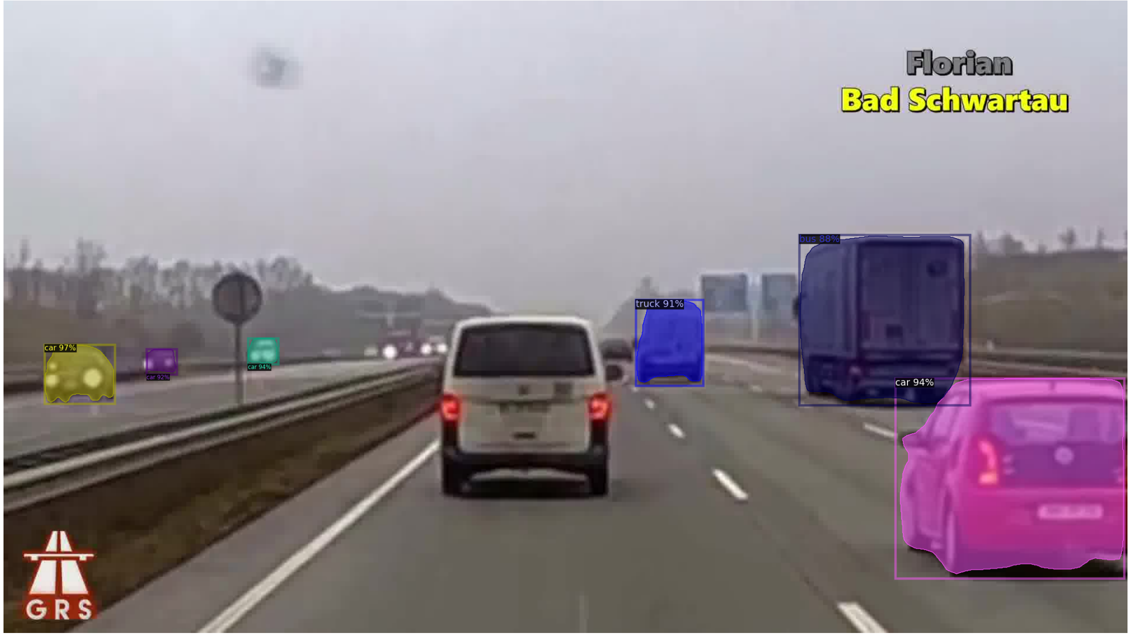
Florian
Bad Schwartau
GRS

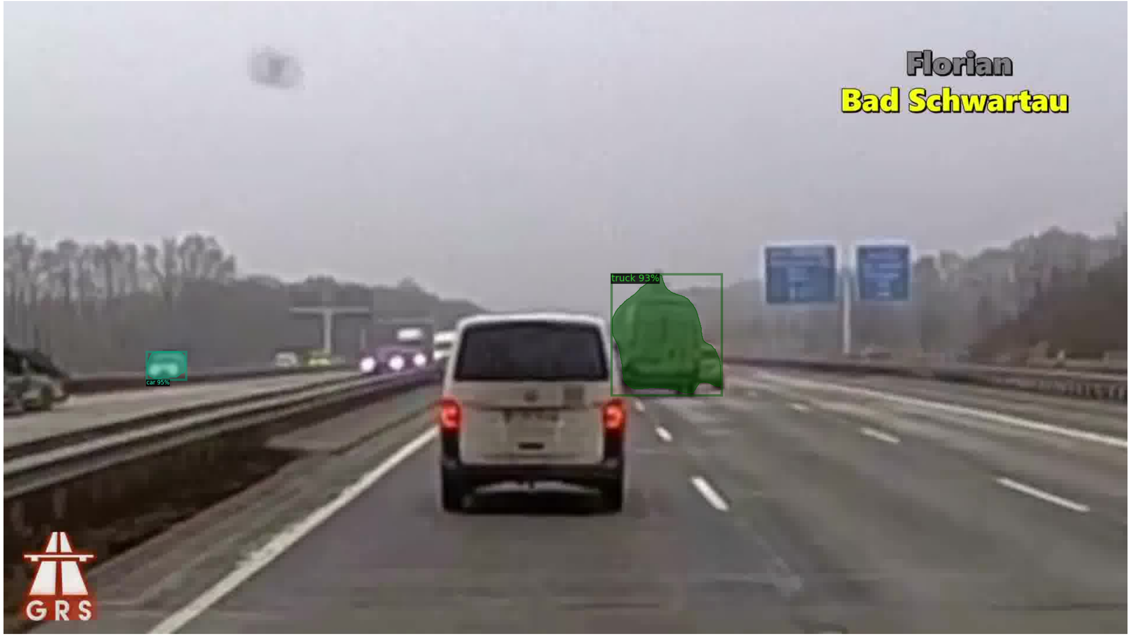
1.1.5 Predicting semantic segmention of loaded images

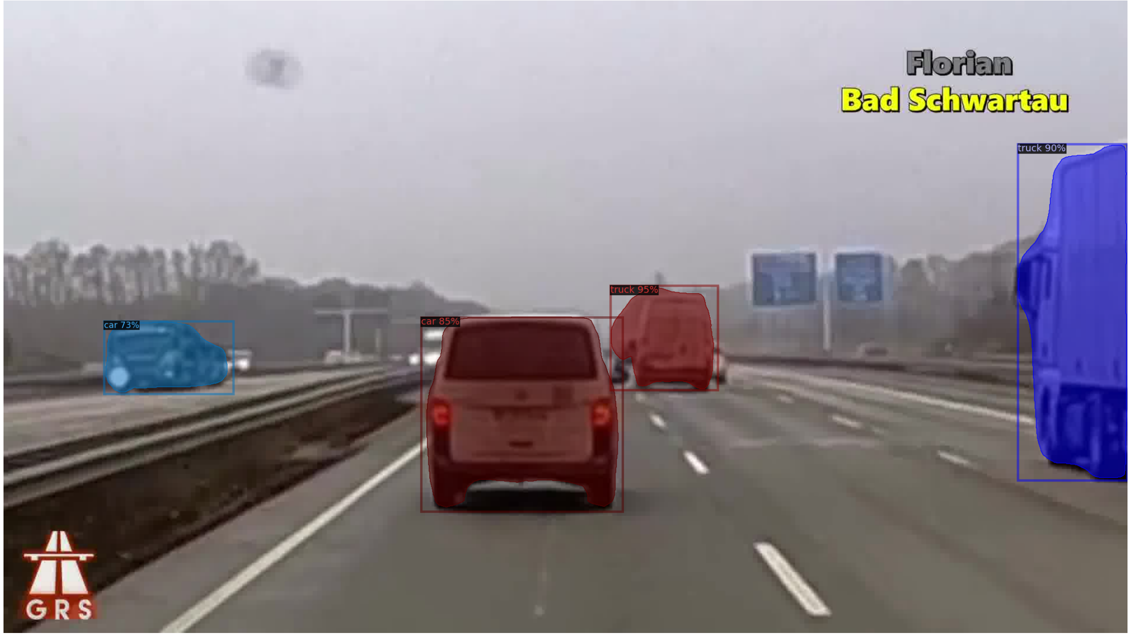
```
In [0]: def semantic_predictor(images):
          output_im_list = []
          for image in images:
            output_im_list.append(seg_predictor(image))
          return output_im_list
        predictions = semantic_predictor(images_list)
```

1.1.6 Visualizing the predictions from a random set of frames to make sure things look correct.

```
In [7]: for indx in random.sample(range(0, 40), 3):
          v = Visualizer(images_list[indx][:, :, ::-1], MetadataCatalog.get(cfg.DATASETS.TRAIN
          v = v.draw_instance_predictions(predictions[indx]["instances"].to("cpu"))
          cv2_imshow(v.get_image()[:, :, ::-1])
```


Florian
Bad Schwartau
car 97%
car 92%
car 94%
truck 91%
bus 88%
car 94%
GRS


Florian
Bad Schwartau
truck 93%
car 95%
GRS



1.2 Part B: Tracking Objects in Pairs of Frames

1.2.1 Computing the matching score

```
In [0]: # Computing Bbox overlap of two predictions
        def bbox_overlap(bbox1,bbox2):
          """
          The bounding box data in coco format is [x,y,h,w]
          x: is the horizontal coordinate of top left
          y: is the vertical coordinate of top left
          h: the lenght of the height of the bbox
          w: the lenght of the width of the bbox

          the model output format is [x1,y1,x2,y2]
          """
          bbox1 = bbox1.pred_boxes[0].tensor[0]
          bbox2 = bbox2.pred_boxes[0].tensor[0]

          x1,y1 = max(bbox1[0], bbox2[0]), max(bbox1[1], bbox2[1])
          x2,y2 = min(bbox1[2], bbox2[2]), min(bbox1[3], bbox2[3])
          if x1 > x2 or y1 > y2: return 0
          return (x1 - x2) * (y1 - y2)

        # Computing the matching score
        def matching_score(pres_frame, next_frame):
          horizontal_dim , vertical_dim =  len(pres_frame),len(next_frame)
          m = np.zeros((horizontal_dim,vertical_dim))
```

```
      for h_index in range(horizontal_dim):
        for v_index in range(vertical_dim):
          if pres_frame[h_index].pred_classes == next_frame[v_index].pred_classes:
            m[h_index][v_index]  = 1
          overlap_value = bbox_overlap(pres_frame[h_index], next_frame[v_index])
          m[h_index][v_index] = m[h_index][v_index] * overlap_value
      mx = np.max(m,axis=1)
      best_match = np.argmax(m, axis=1)
      best_match[mx==0] = -99
      return best_match,m
```

```
In [0]: # tracking in the whole frames
        tracking_list = []
        for index in range(len(predictions)-1):
          temp_track ,m= matching_score(predictions[index]["instances"],predictions[1+index]["
          tracking_list.append(temp_track)
```

1.2.2 Color-coding tracks

```
In [0]: def generate_colors(n):
          """
          Take an integer n and it egenerates n RGB colors
          those RGB values range between 0 and 1
          """
          ret = []
          r,g,b = 158,39,248
          step = 256 / n
          for i in range(n):
            r += step
            g += step
            b += step
            r = int(r) % 256
            g = int(g) % 256
            b = int(b) % 256
            ret.append((r/255,g/255,b/255))
          return ret
```

1.2.3 Object tracking in a Video

```
In [0]: init_pred_classes = predictions[0]["instances"].pred_classes
        present_colors = generate_colors(len(init_pred_classes))
        class_names = ["Track "+str(i) for i in range(len(init_pred_classes))]
        global_class_name_counter = len(class_names)
        res = []
        num_frames = len(images_list)
        for indx in range(num_frames):
              img = images_list[indx]
              v = Visualizer(img[:, :, ::-1], MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scal
           for cidx, bbox in enumerate(predictions[indx]["instances"].pred_boxes.tensor):
                text_pos = (bbox[0],bbox[1])
                v.draw_box(bbox,edge_color=present_colors[cidx],alpha=0.9)
                v.draw_text( class_names[cidx],text_pos, color=present_colors[cidx],for
           img = v.get_output().get_image()[:, :, ::-1]
           output = img
           res.append(output)

           if indx < (num_frames - 1):
               init_pred_classes = predictions[indx+1]["instances"].pred_classes
               present_colors = generate_colors(len(init_pred_classes))
               new_class_names = [""] * (len(init_pred_classes))
               for indx,idx in enumerate(tracking_list[indx]):
                 if idx > 0:
                   new_class_names[idx] = class_names[indx]
               for i, name in enumerate(new_class_names):
                 if name=="":
                   new_class_names[i]="Track "+str(global_class_name_counter)
                   global_class_name_counter+=1
               class_names=new_class_names
```

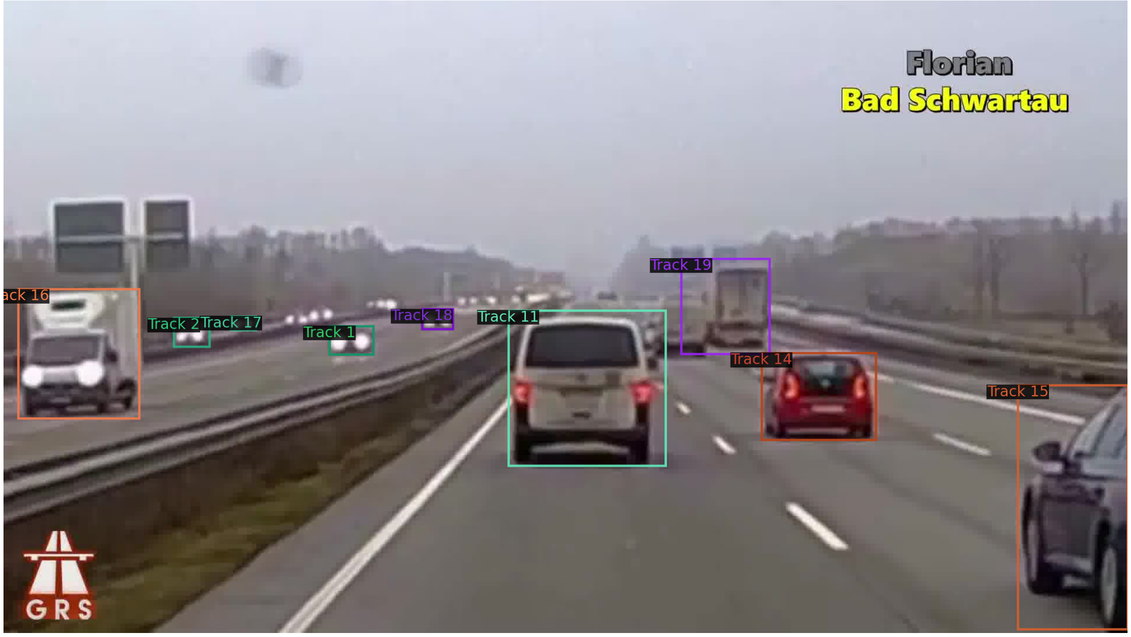
1.2.4 Pick a clip required by specifying the start and end frames

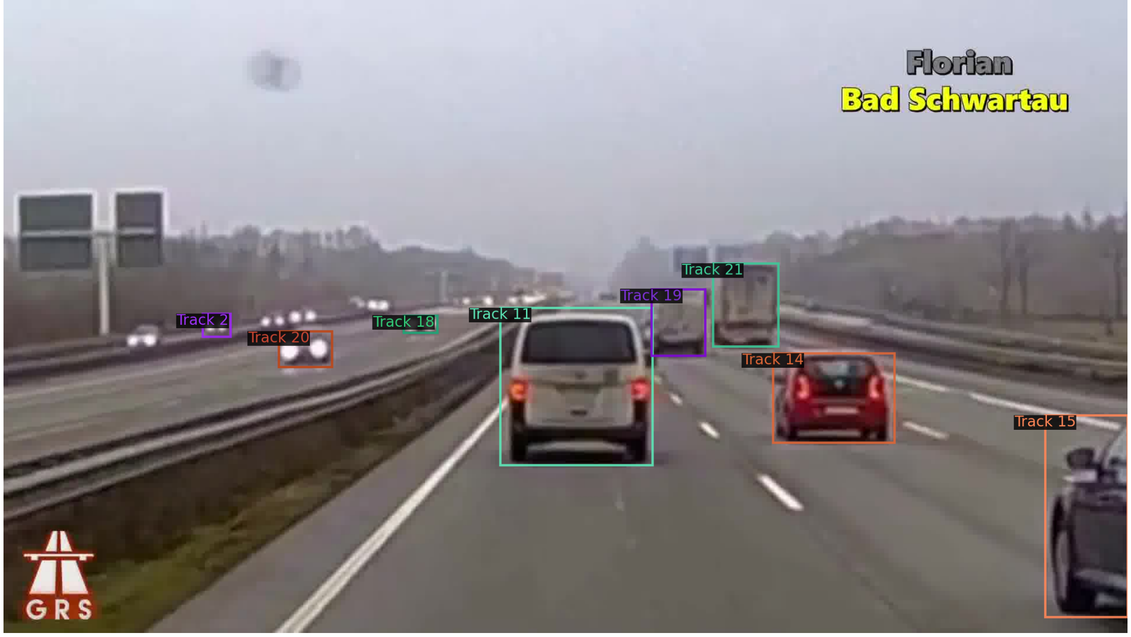
```
In [0]: def pick_clip_in_video(frames,start_frame=0,end_frame=10):
           """
           From a list of frames, returns the range of specified frames
           """
           res = []
           if start_frame < 0 or end_frame >= len(frames):
             start_frame = 0
             end_frame = len(frames)
             print('wrong frame indexes!!!!!.....')
           while start_frame < end_frame:
                 res.append(frames[start_frame])
                 start_frame += 1
           return res
```

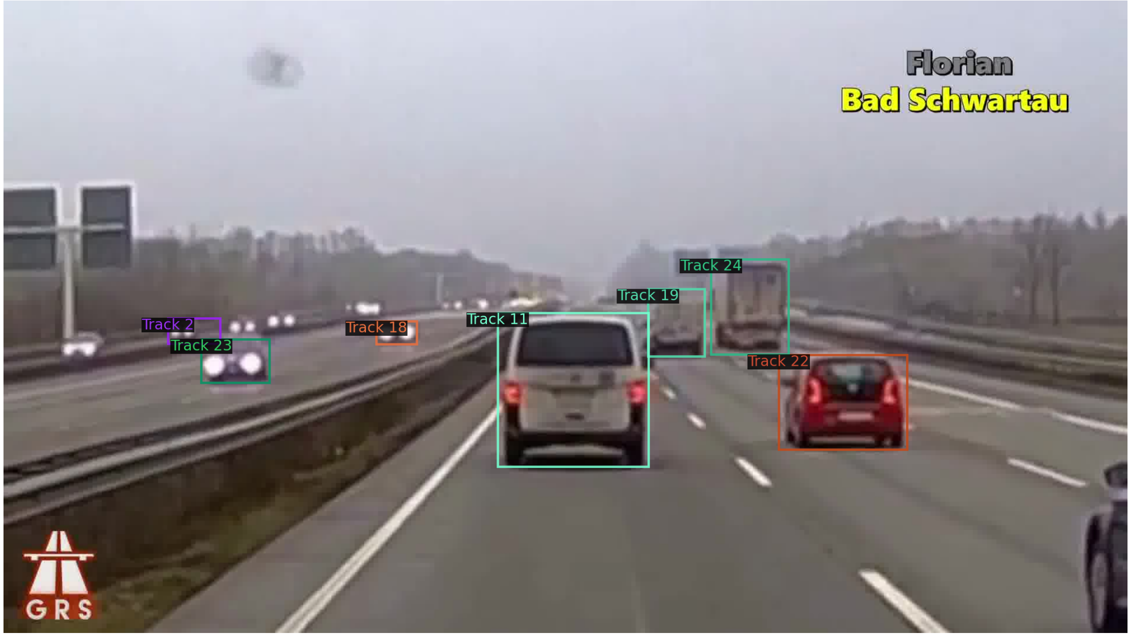
```
In [0]: ans = pick_clip_in_video(res,start_frame=2,end_frame=12)
```

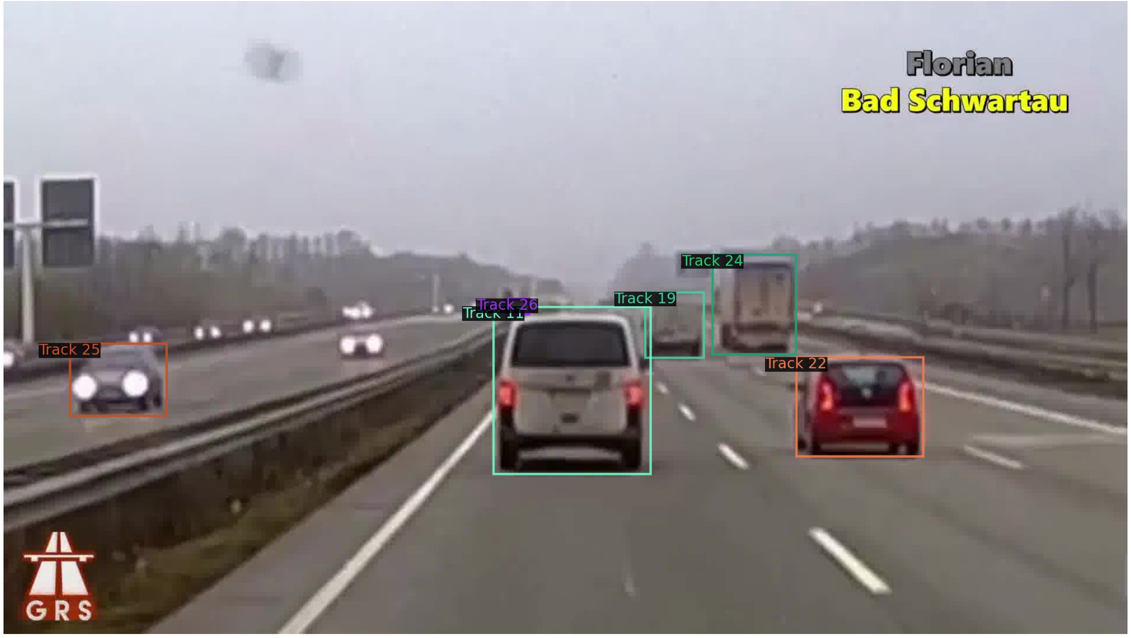
```
In [14]: print(len(ans))
         for i in range(len(ans)):
           cv2_imshow(ans[i])
```

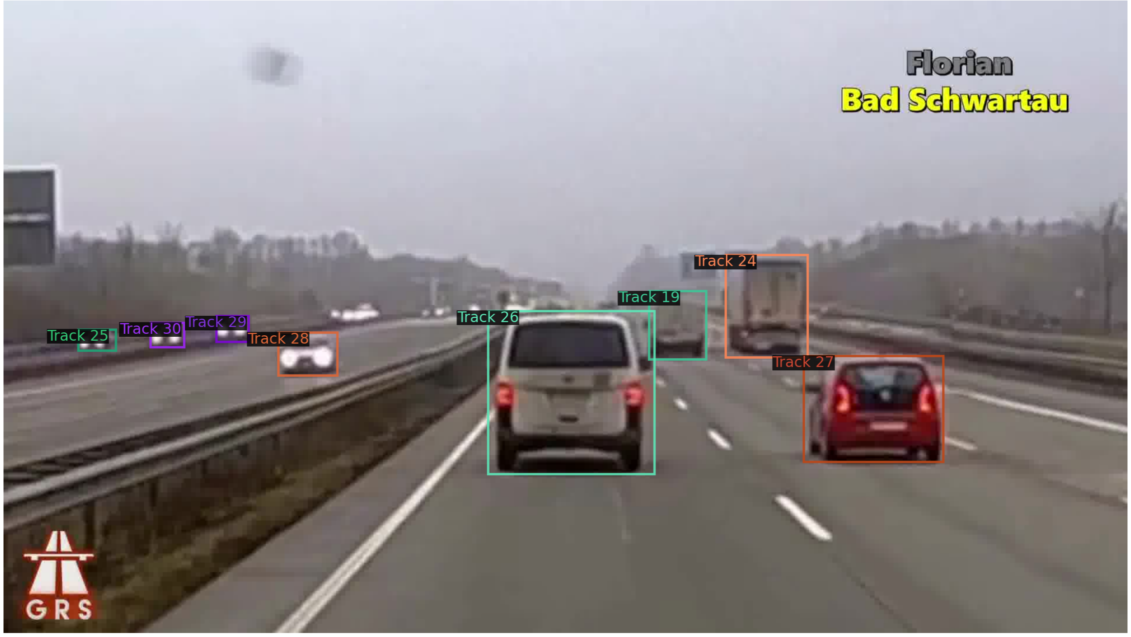
```
10
```

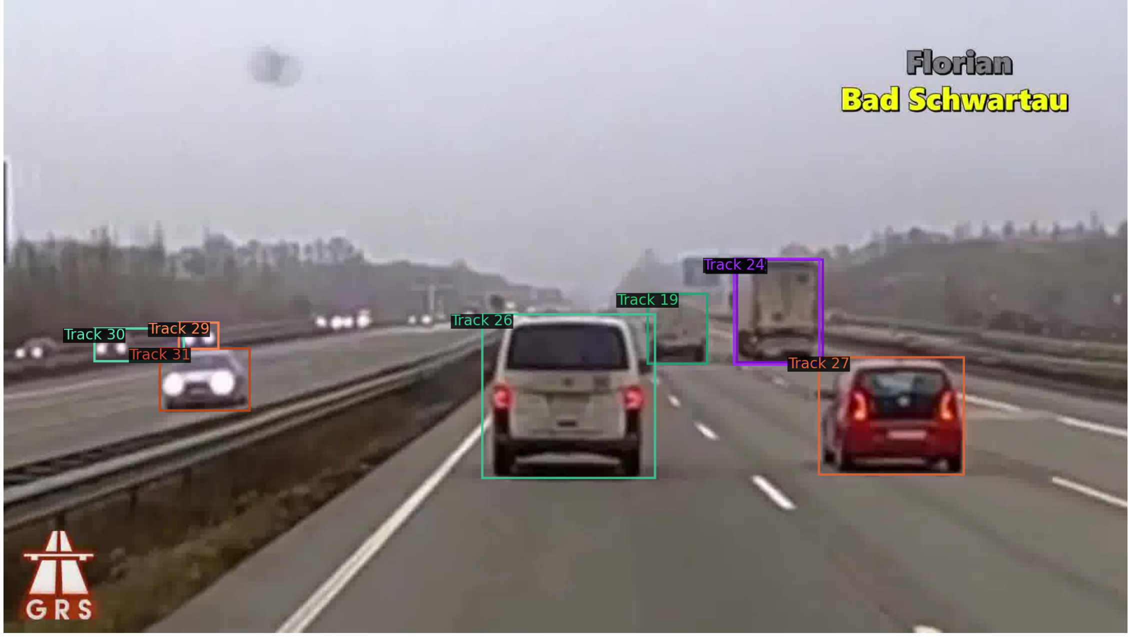

Florian
Bad Schwartau
ack 16
Track 2
Track 17
Track 1
Track 18
Track 11
Track 19
Track 14
Track 15
GRS

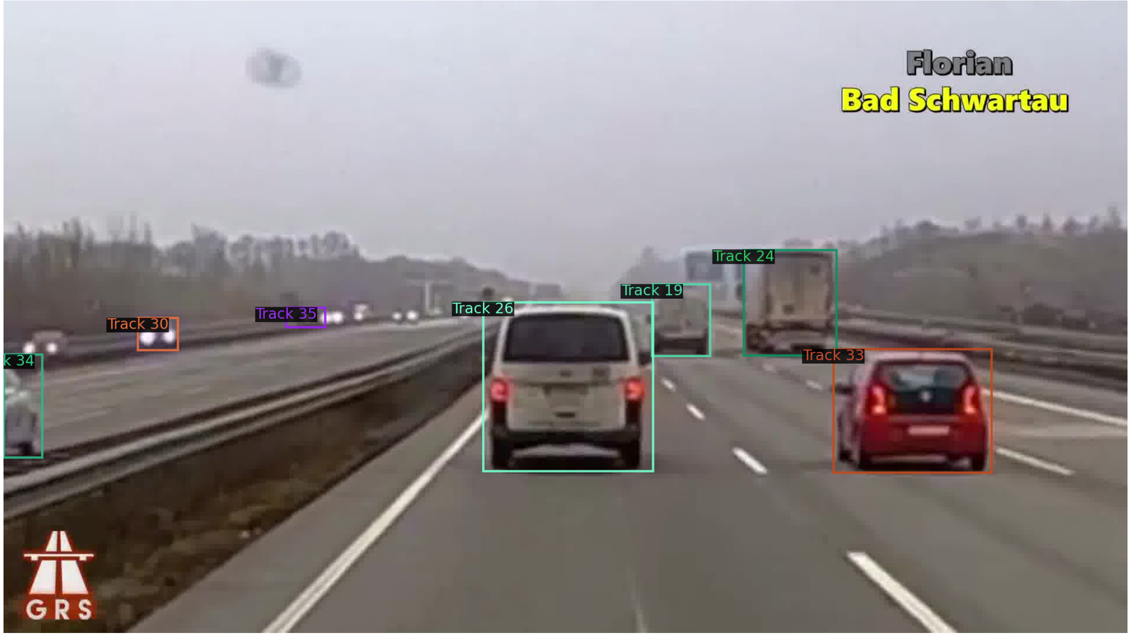

Florian
Bad Schwartau
Track 21
Track 19
Track 2
Track 20
Track 18
Track 11
Track 14
Track 15
GRS

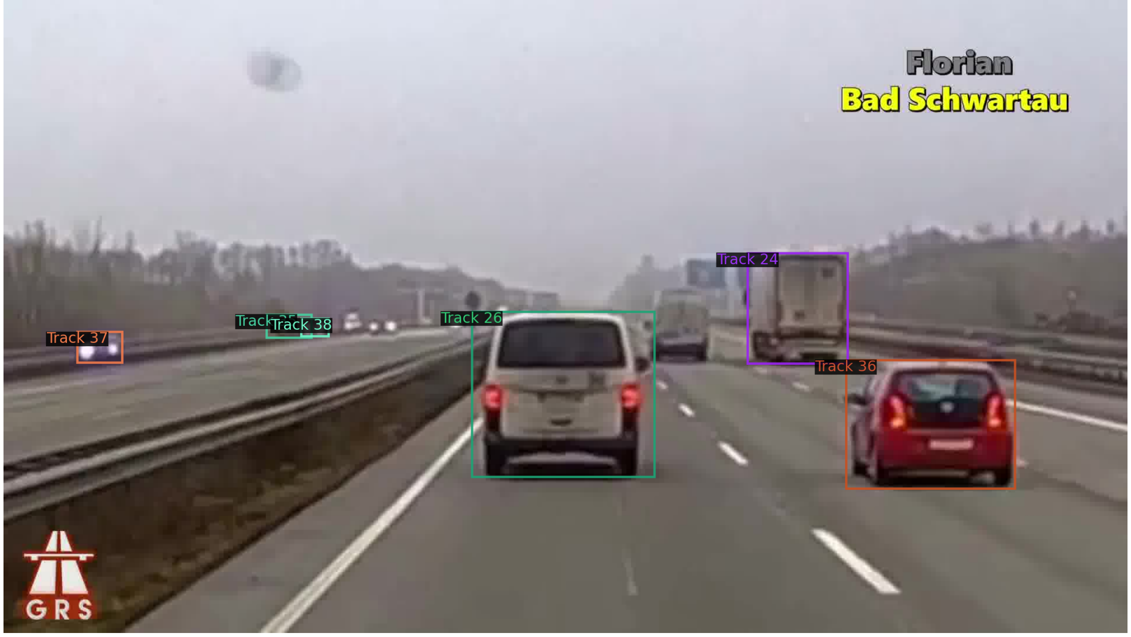

Florian
Bad Schwartau
Track 24
Track 19
Track 11
Track 2
Track 23
Track 18
Track 22
GRS

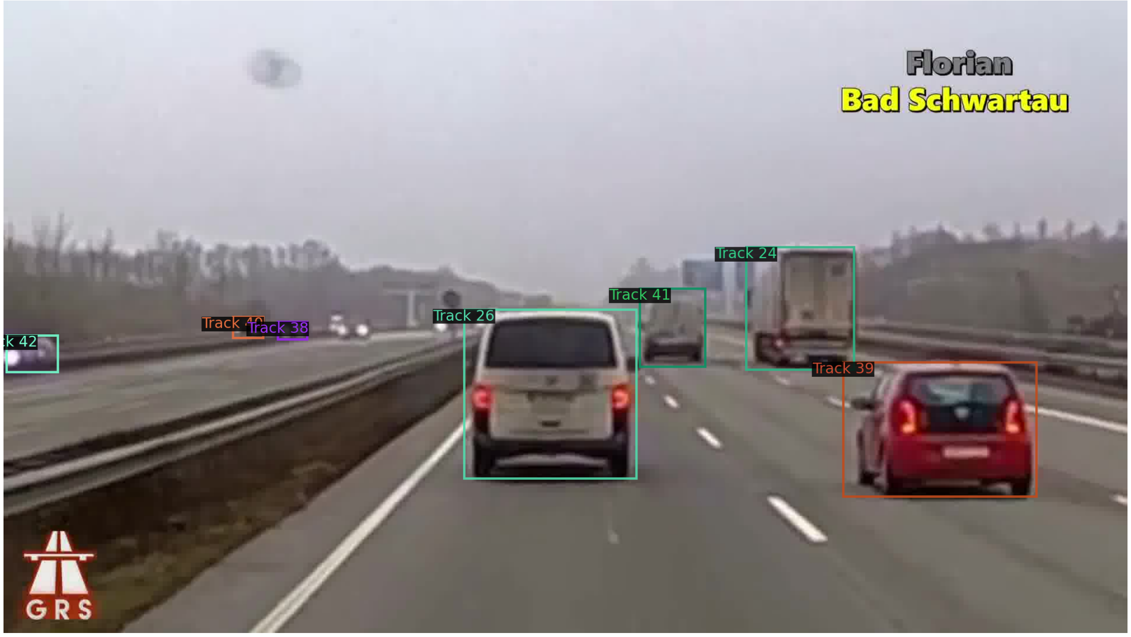

Florian
Bad Schwartau
Track 24
Track 26
Track 11
Track 19
Track 25
Track 22
GRS

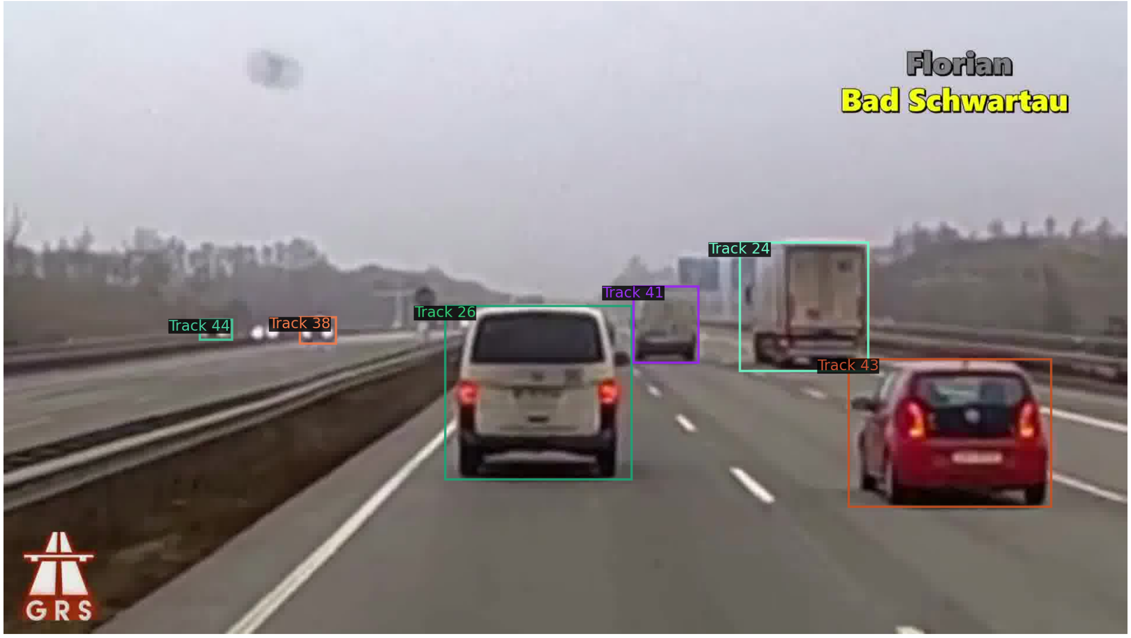

Florian
Bad Schwartau
Track 24
Track 19
Track 26
Track 27
Track 25
Track 30
Track 29
Track 28
GRS


Florian
Bad Schwartau
Track 24
Track 19
Track 26
Track 27
Track 30
Track 29
Track 31
GRS


Florian
Bad Schwartau
Track 24
Track 19
Track 26
Track 35
Track 30
k 34
Track 33
GRS


Florian
Bad Schwartau
Track 24
Track 26
Track 38
Track 37
Track 36
GRS





1.2.5 Create a video and download it

```
In [0]: def ouput_video_maker(images_list,file_name = 'tracker',fps = 15):
            """
            This function creates video
```

```
        @parameters:
            images_list: This is a list of annotated images with appropriate color codin
            file_name : The name that you want to save your video as
            fps: frops per second, this define speed of your frames
        @output:
            Video, and Download the video
        """
        file_name = file_name + '.avi'
        size = (len(images_list[0][0]),len(images_list[0]))
        out = cv2.VideoWriter(file_name, cv2.VideoWriter_fourcc(*'DIVX'),fps,size)
        print('Creating the file')
        for frame in images_list:
          out.write(frame)
        out.release()
        print('Downloading the file')
        files.download(file_name)
        return 'Done'
```

```
In [17]: ouput_video_maker(images_list = ans,file_name = 'object_tracking',fps = 2)
```

```
Creating the file
Downloading the file
```

```
Out[17]: 'Done'
```

```
In [0]: #visualize tracking in the whole frames
        for i in range(40):
          cv2_imshow(res[i])
```

1.2.6 The end!!!!

```
In [0]:
```